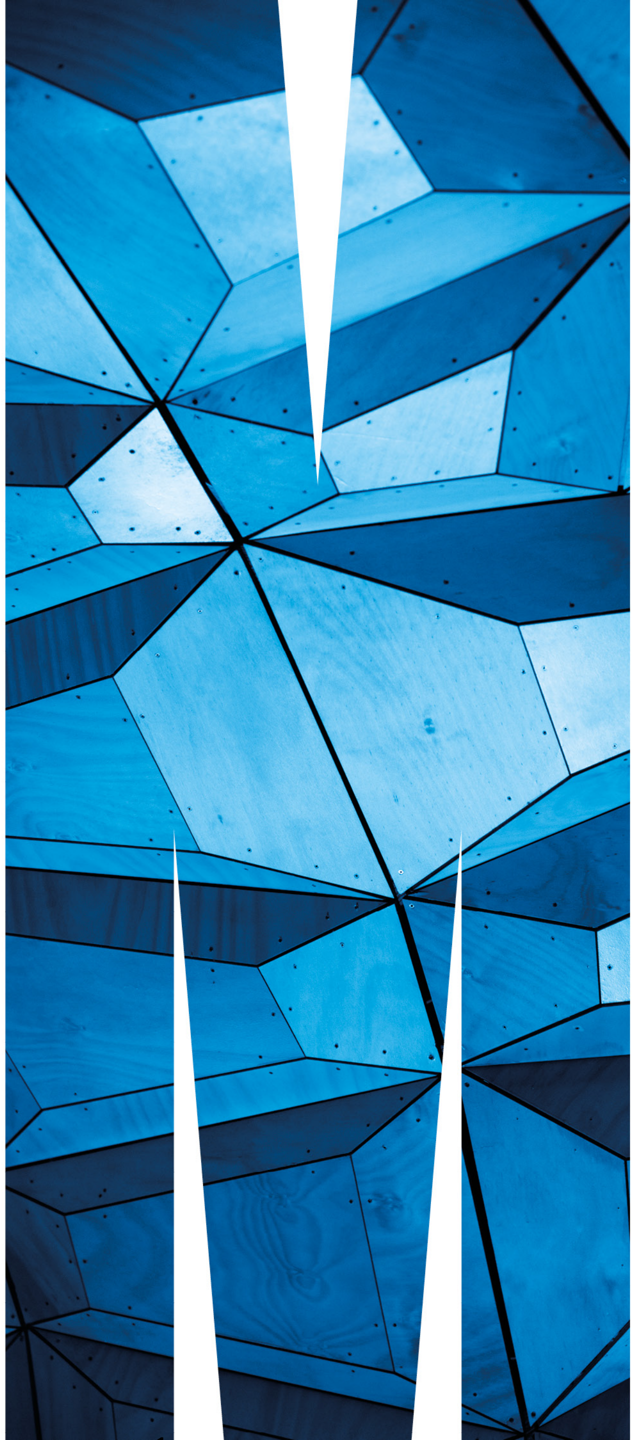
MONASH University

Rate Monotonic Algorithm (RMA) for Periodic Tasks



Also called RMS (Rate Monotonic Scheduling)

Clive Maynard 2022

A good reference for RMA Scheduling is:
Klein et al "A Practitioner's Handbook for Real-time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems"
Klewer

MONASH University

We need a way to guarantee that a task set can be scheduled successfully.

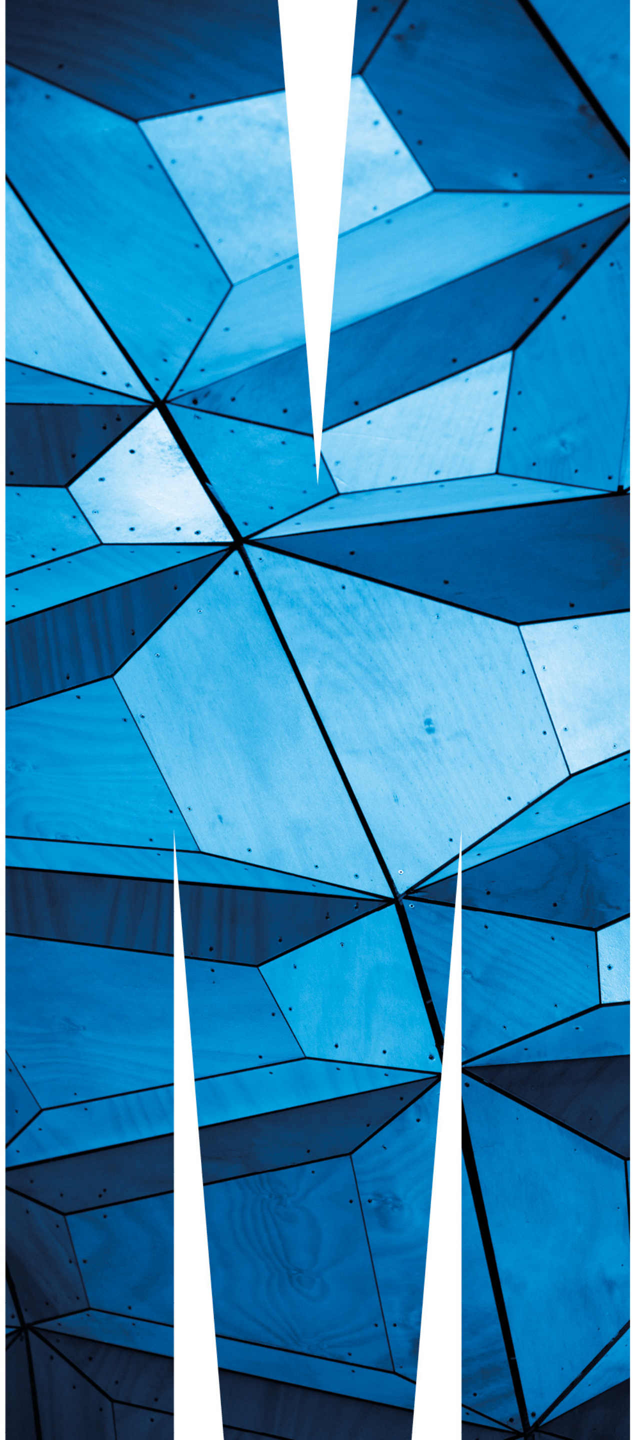
Each task meeting its own deadline.

Success is not only completing a task but doing it within specified time constraints.

The Rate Monotonic Algorithm was the first to provide a guarantee for scheduling based on periodic execution requirements.

Liu & Layland 1973

Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment

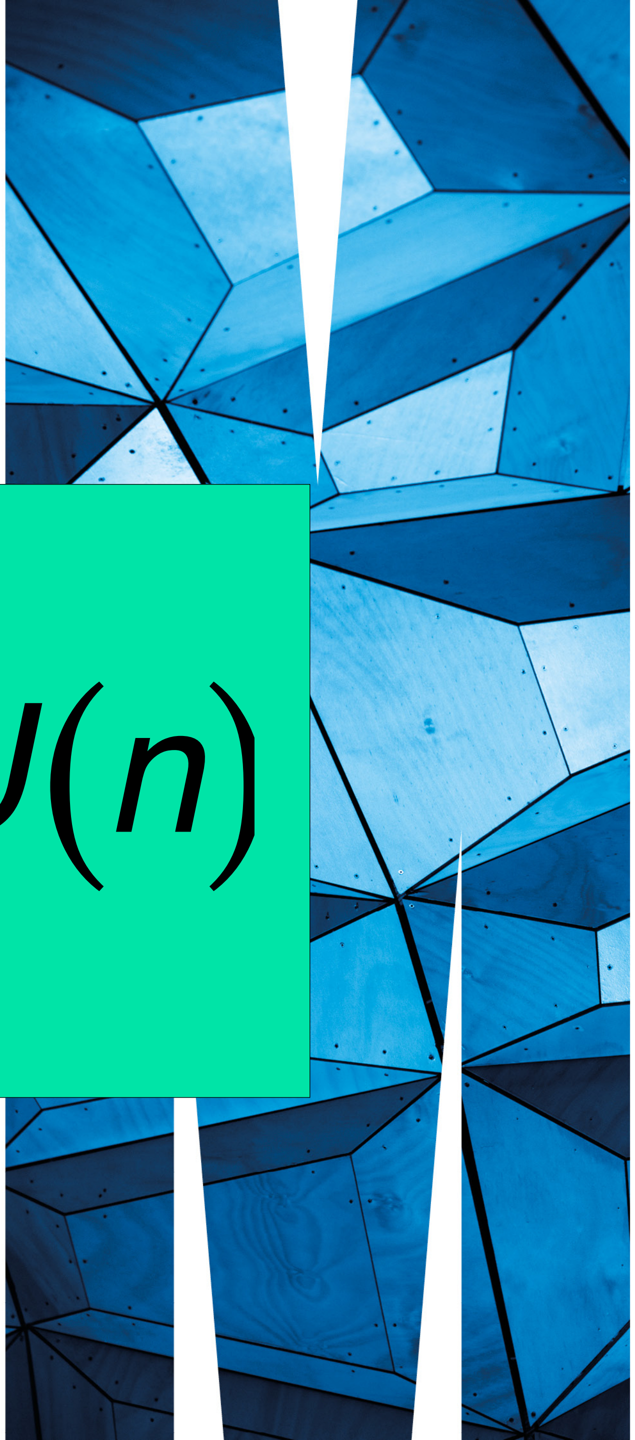


MONASH University

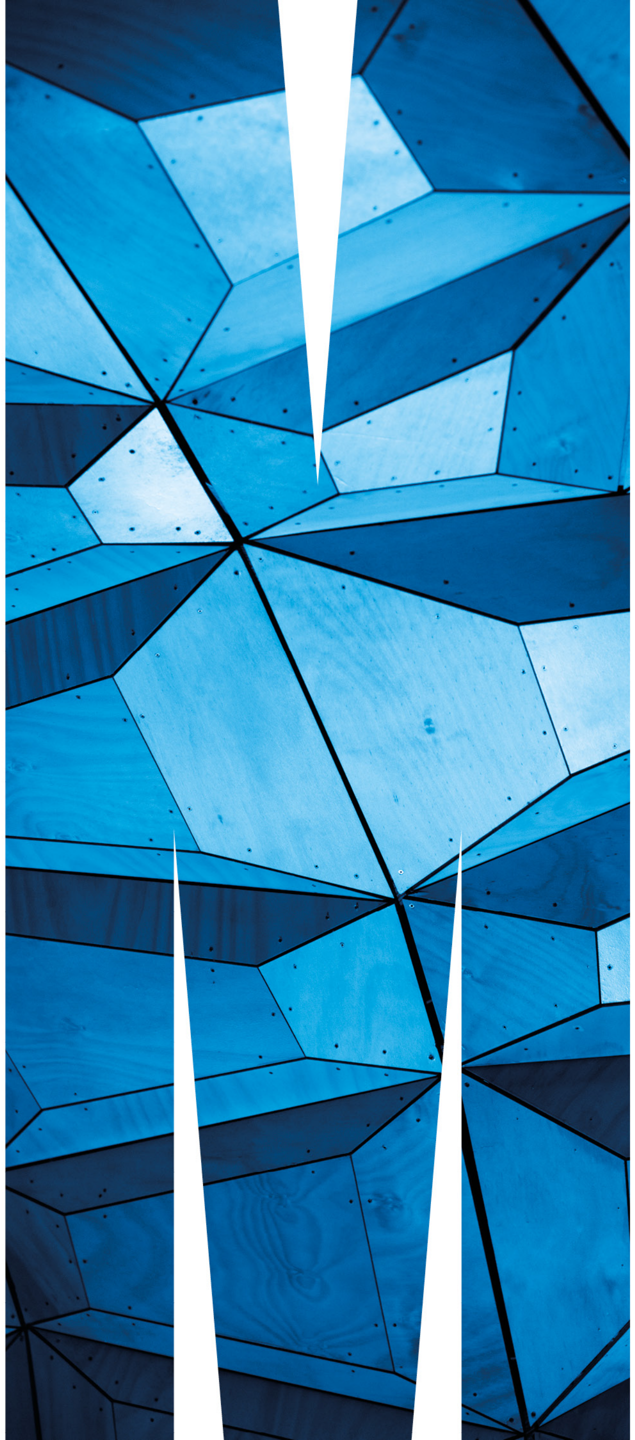
Theorem 1:

$$\sum_{j=1}^{n}\left(\frac{C_k}{T_k}\right) \le n\left(2^{\frac{1}{n}} - 1\right) = U(n)$$

C is execution time, T is period and n the number of tasks



Conditions



The requests for all tasks for which hard deadlines exist are periodic.

Deadlines consist of run-ability constraints only - i.e. each task must be completed before the next request for it occurs.

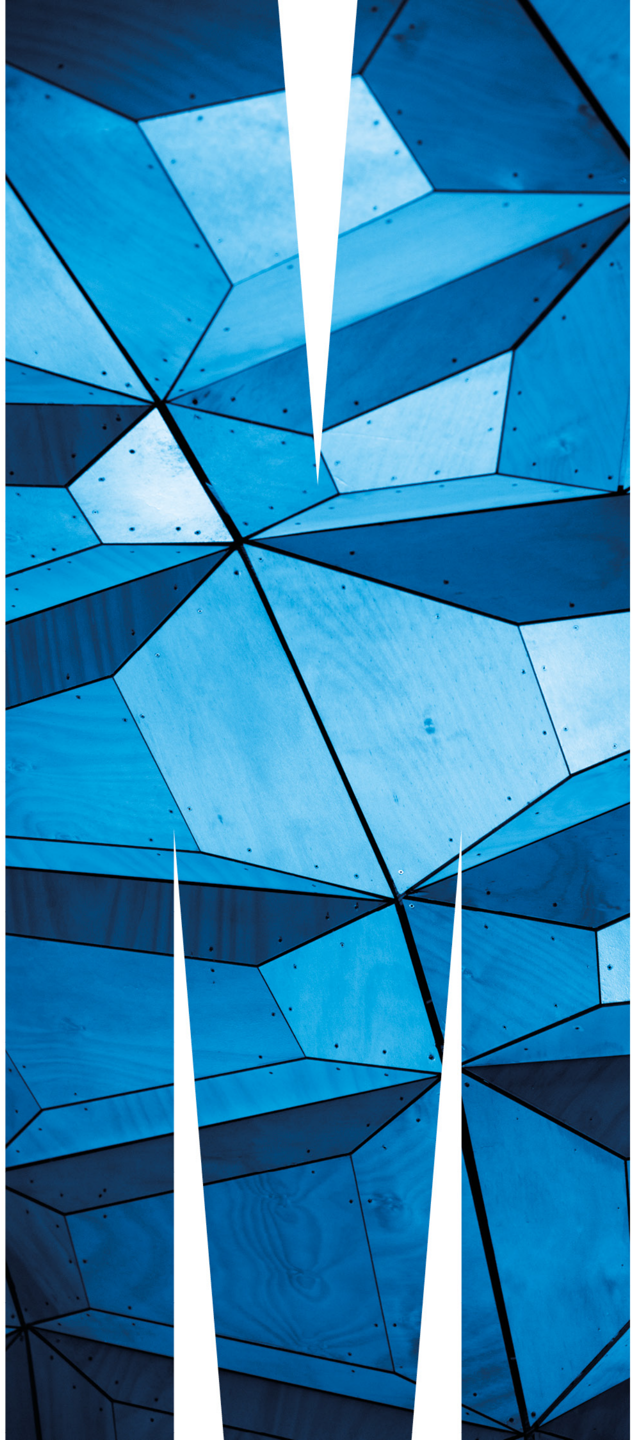
The tasks are independent (in that requests for a certain task do not depend on the initiation or completion of requests for other tasks).

Run-time for each task is constant for that task and does not vary with time.

Any nonperiodic tasks in the system are special; they are initialization or failure recovery routines;

A critical instant for any task occurs whenever the task is requested simultaneously with requests for all higher priority tasks.

The critical instant for a task set occurs when all tasks in the set are scheduled at the same instant.



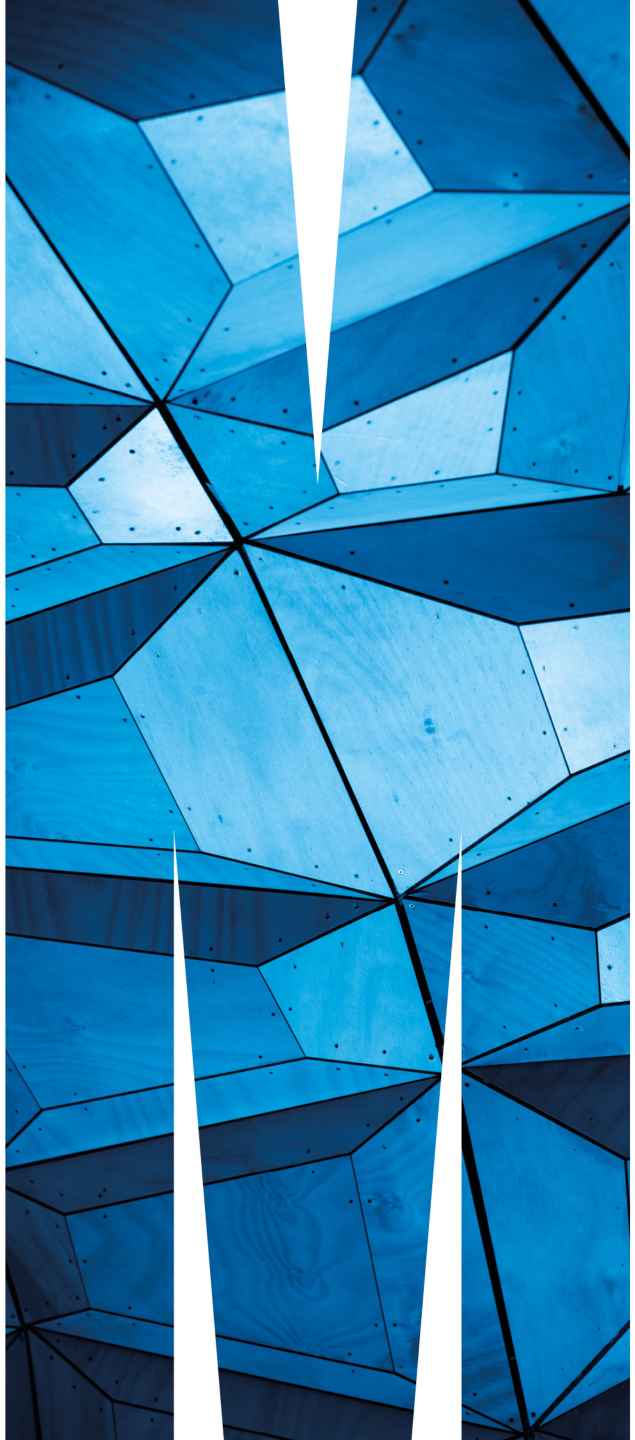
Theorem 1 offers a sufficient (worst case) condition that characterises schedulability of a task set under the RMA.

The bound converges to 69% (In 2) as the number of tasks approaches infinity.

This theorem offers a guarantee, for a preemptive scheduler using the RMA conditions, that provided the utilisation is less than U(n) the tasks can be scheduled.

Provided the total utilisation is less than the bound RMA guarantees success.

U(1) = 1.0
U(2) = 0.828
U(3) = 0.779
U(4) = 0.756
U(5) = 0.743
U(6) = 0.734
U(7) = 0.728
U(8) = 0.724
U(9) = 0.720

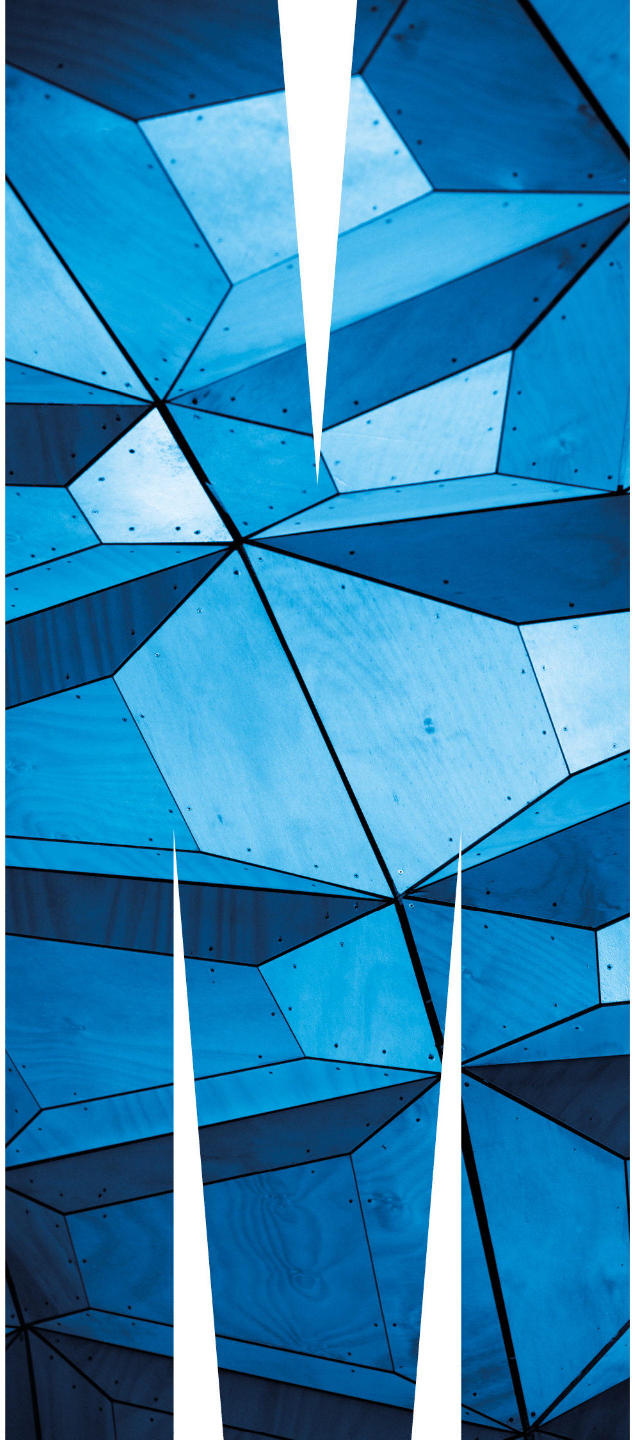


MONASH University

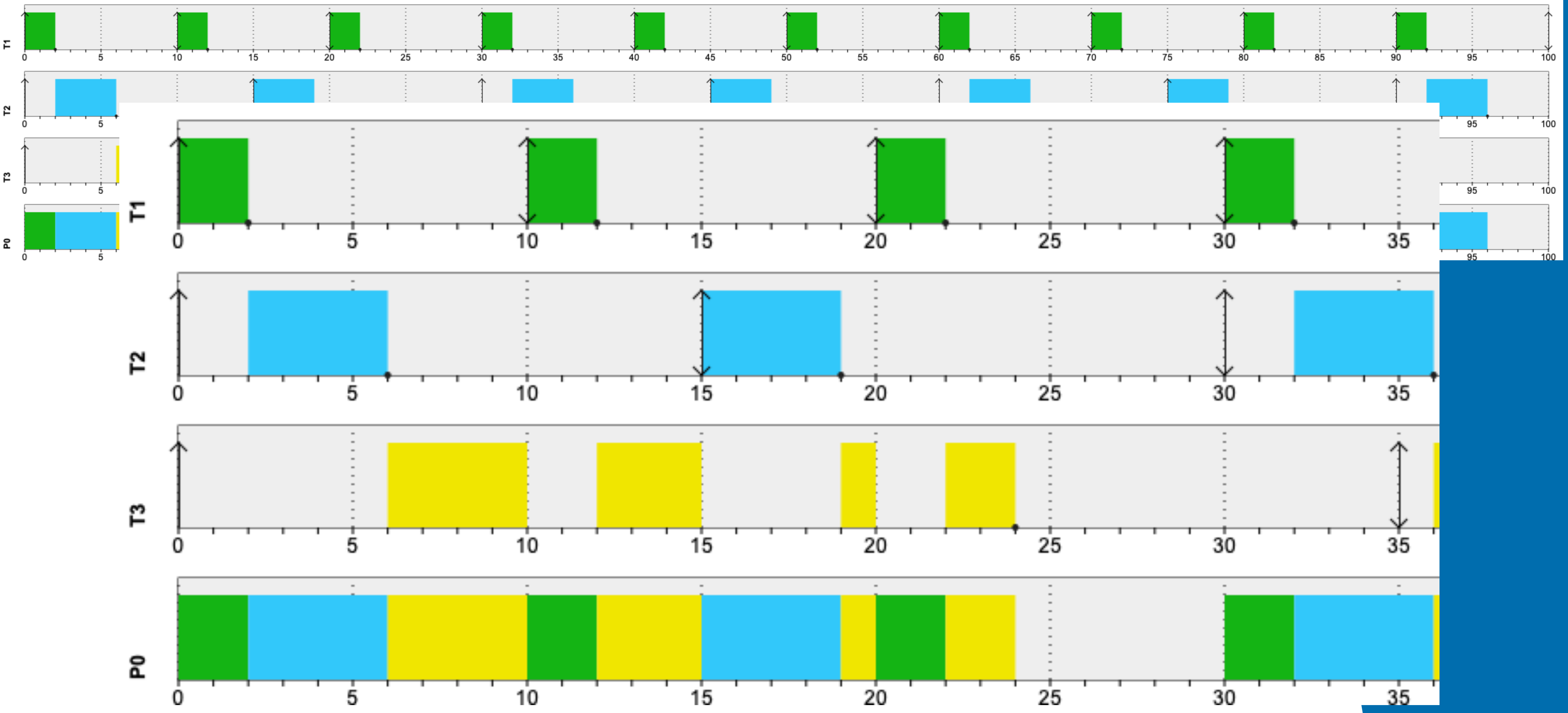
Example:

Task #	C	T	U(n)	Sum of U(n)
1	20	100	0.2	0.2
2	40	150	0.267	0.467
3	100	350	0.286	0.753

The bound for n=3 is 0.779 so the set is schedulable.



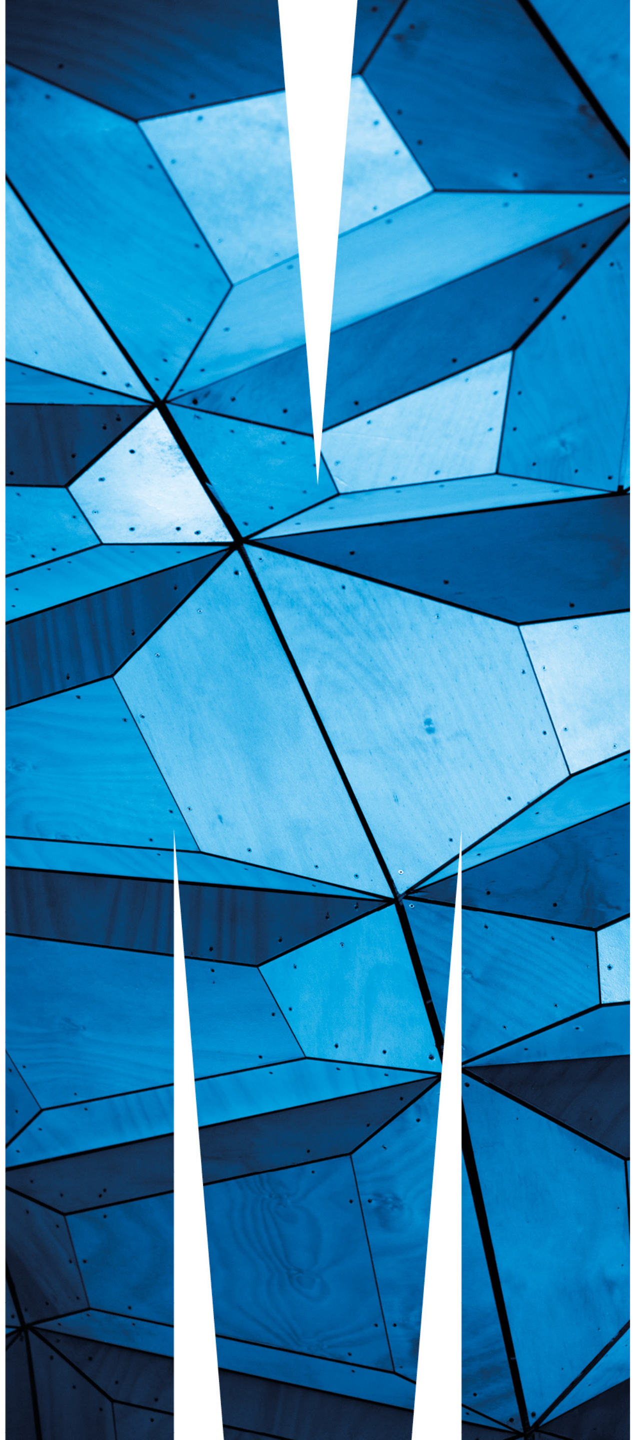
Example running under SimSo (scaled down by factor of 10)



MONASH University

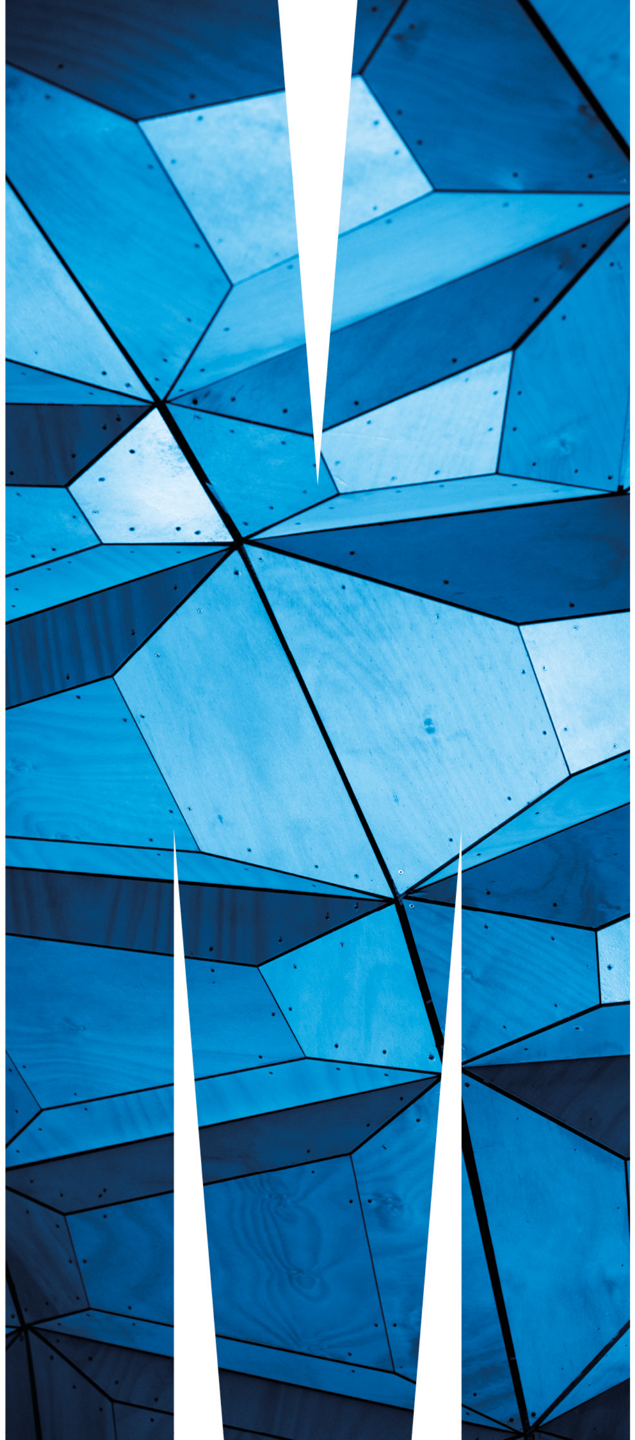
The bound given by Theorem 1 is very pessimistic as the worst case task set is contrived and very unlikely to be encountered in practice.

For a randomly chosen task set the likely bound is 88% but for design you cannot assume this.

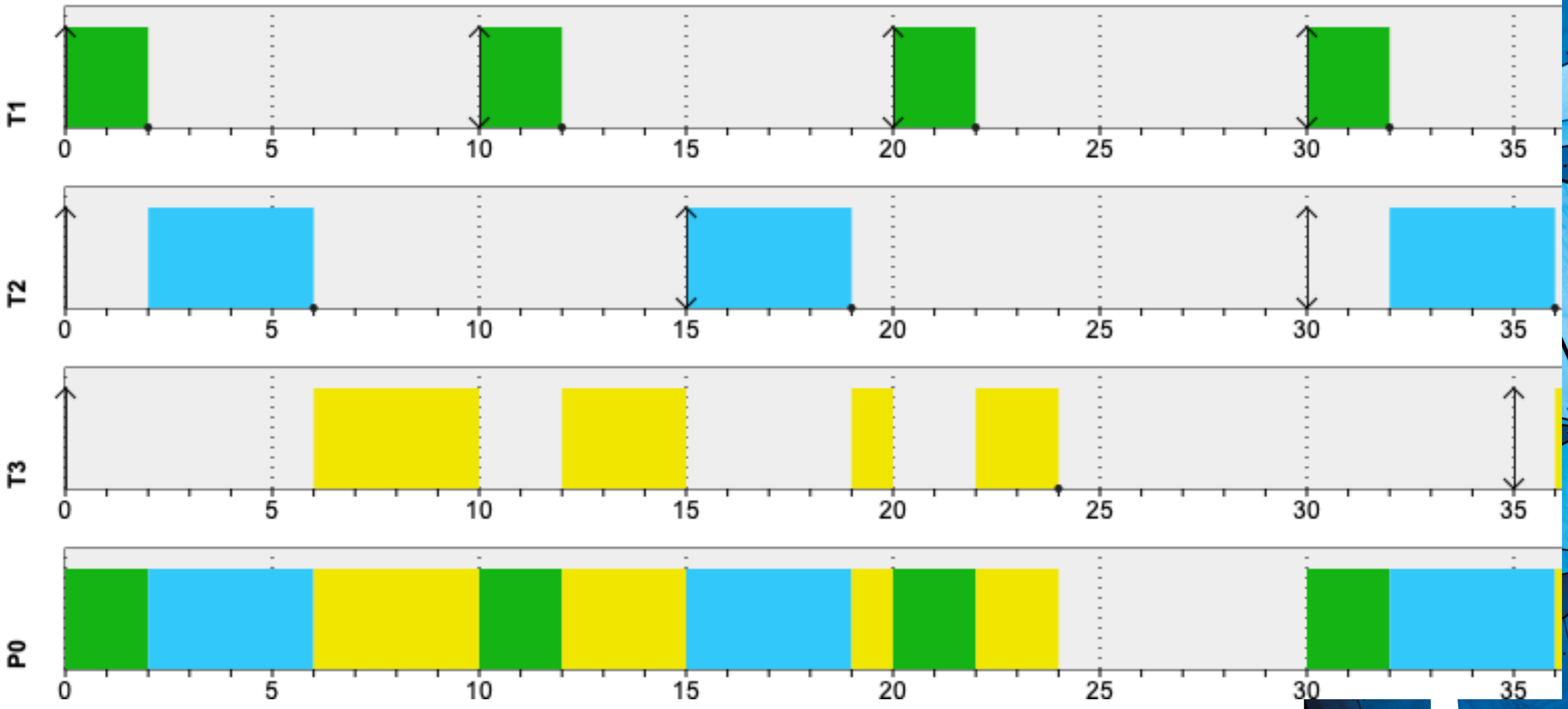


Theorem 2:

For a set of independent periodic tasks, if each task meets its first deadline when all tasks are started at the same time, then the deadlines will always be met for any combination of start times.



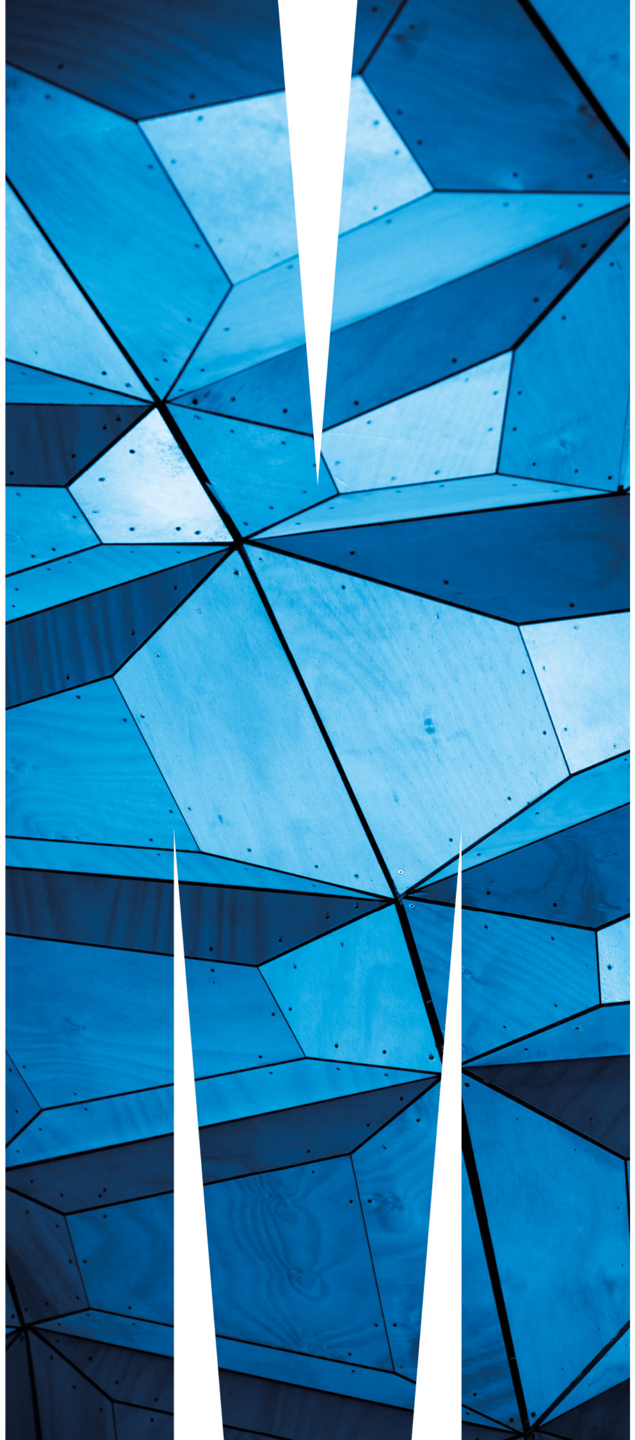
Example (again)



MONASH University

Example…….continued

A better but more compute intensive algorithm is used for Task 1 which raises its execution time to 40 units.

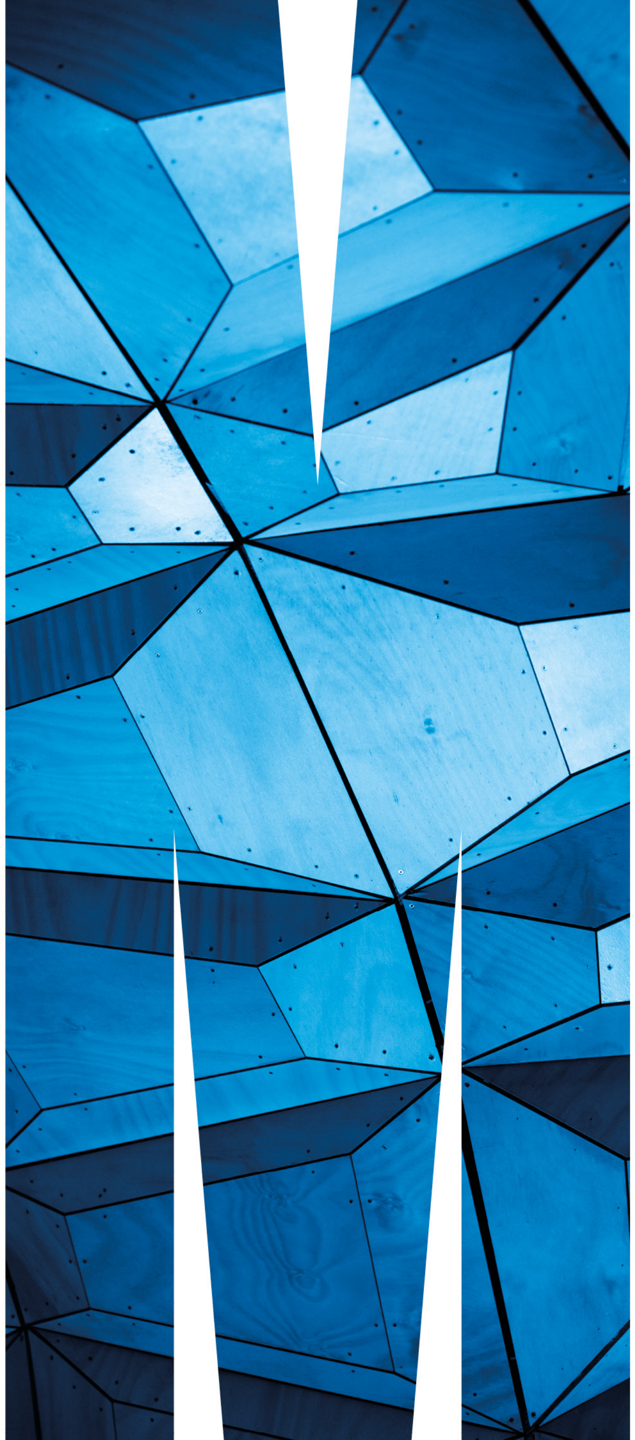


MONASH University

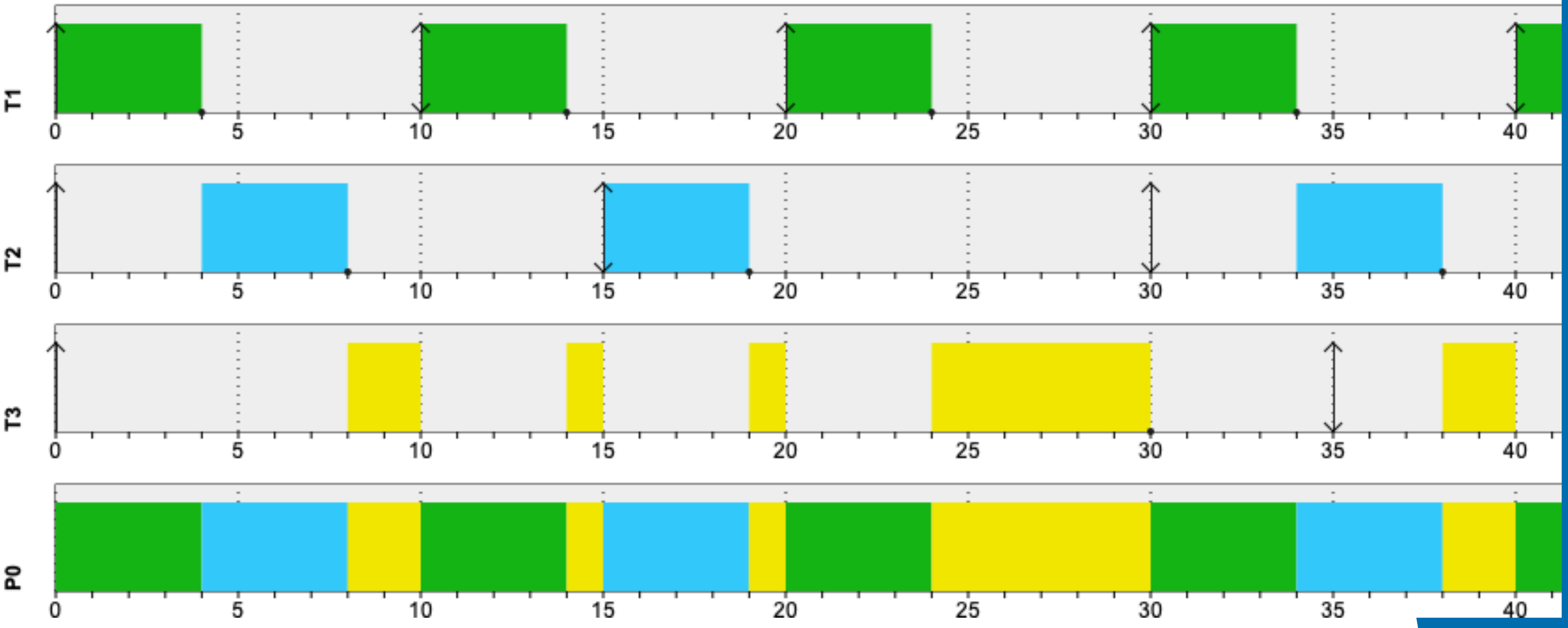
Example:

Task #	C	T	U(n)	Sum of U(n)
1	40	100	0.4	0.4
2	40	150	0.267	0.667
3	100	350	0.286	0.953

The bound for n=3 is 0.779 so the set is not schedulable by Theorem 1.



Example extended (SimSo - time divided by 10).

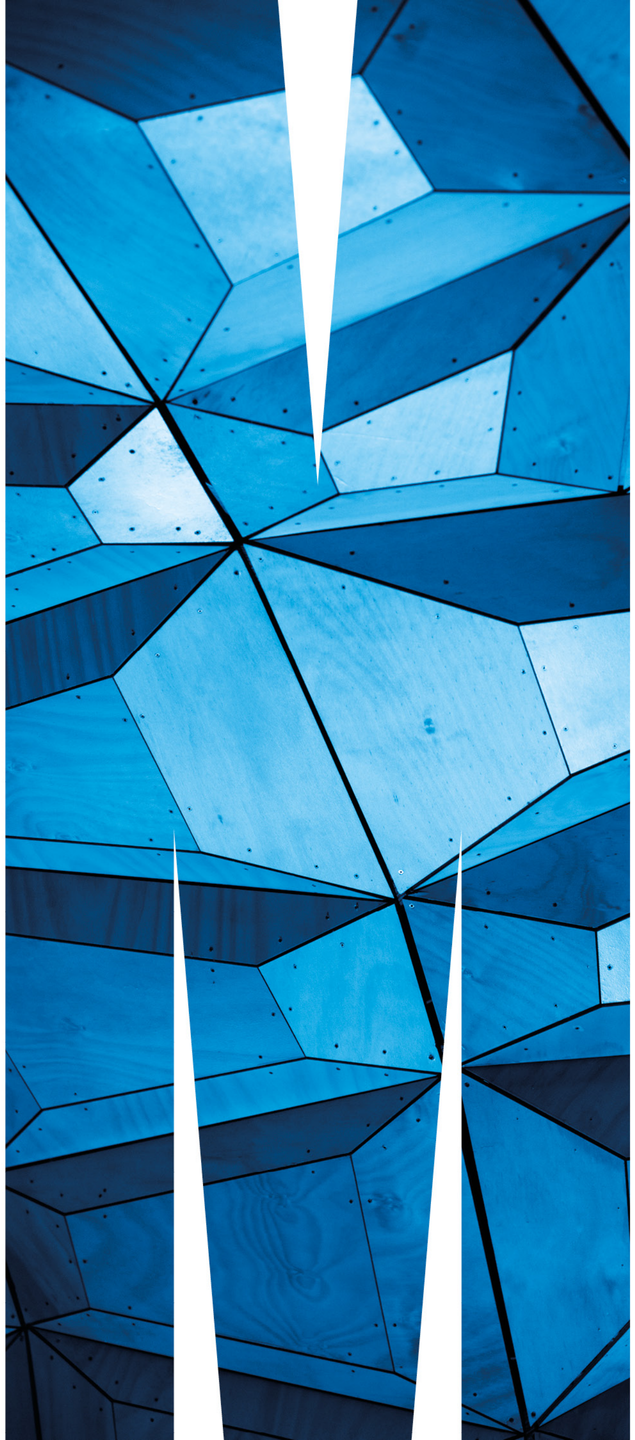


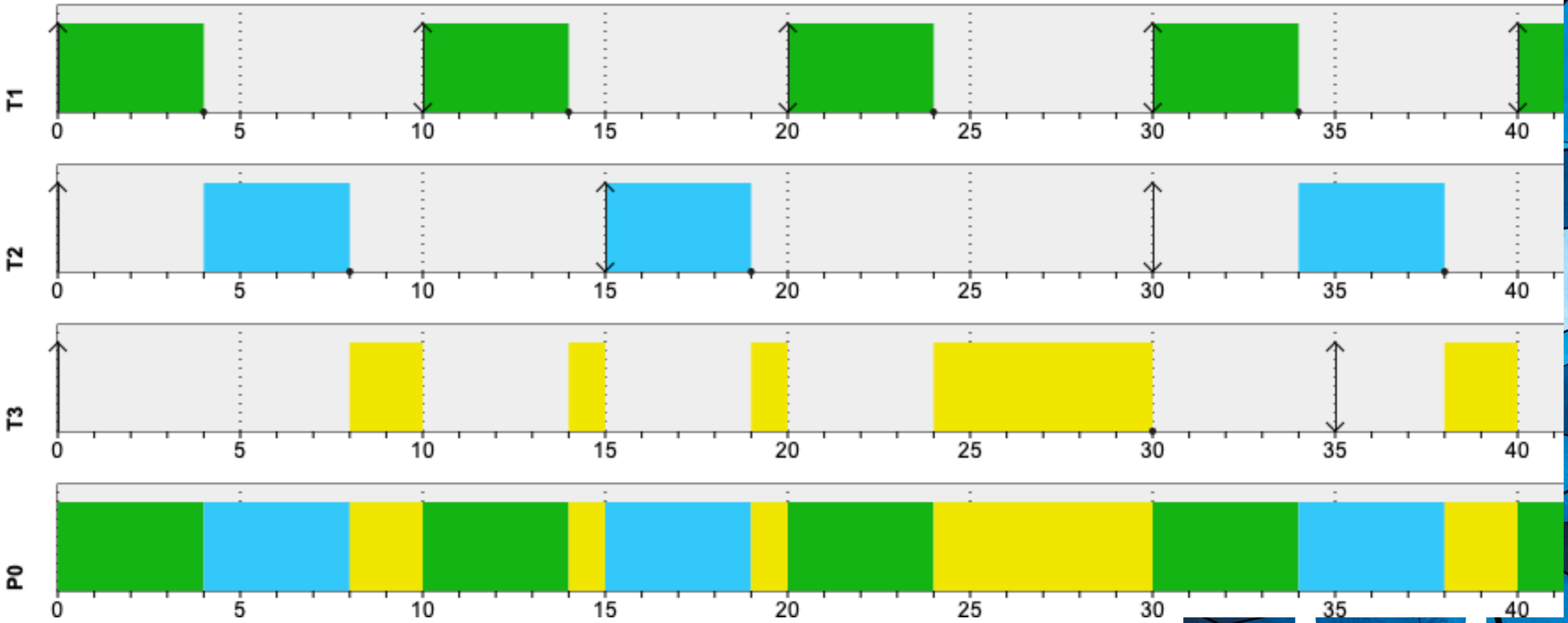
MONASH University

The Utilisation for the first two tasks is 0.667 which is well inside the bound from Theorem 1 so they will meet their deadlines.

Theorem 2 requires that the summation of all execution for all tasks at all the scheduling points be checked upto the deadline for the task under investigation.

If all tasks meet their deadlines for at least one checking point, the task set is schedulable.





Graphically it can be seen that we meet the deadline for task 3 (at time 30)

MONASH University

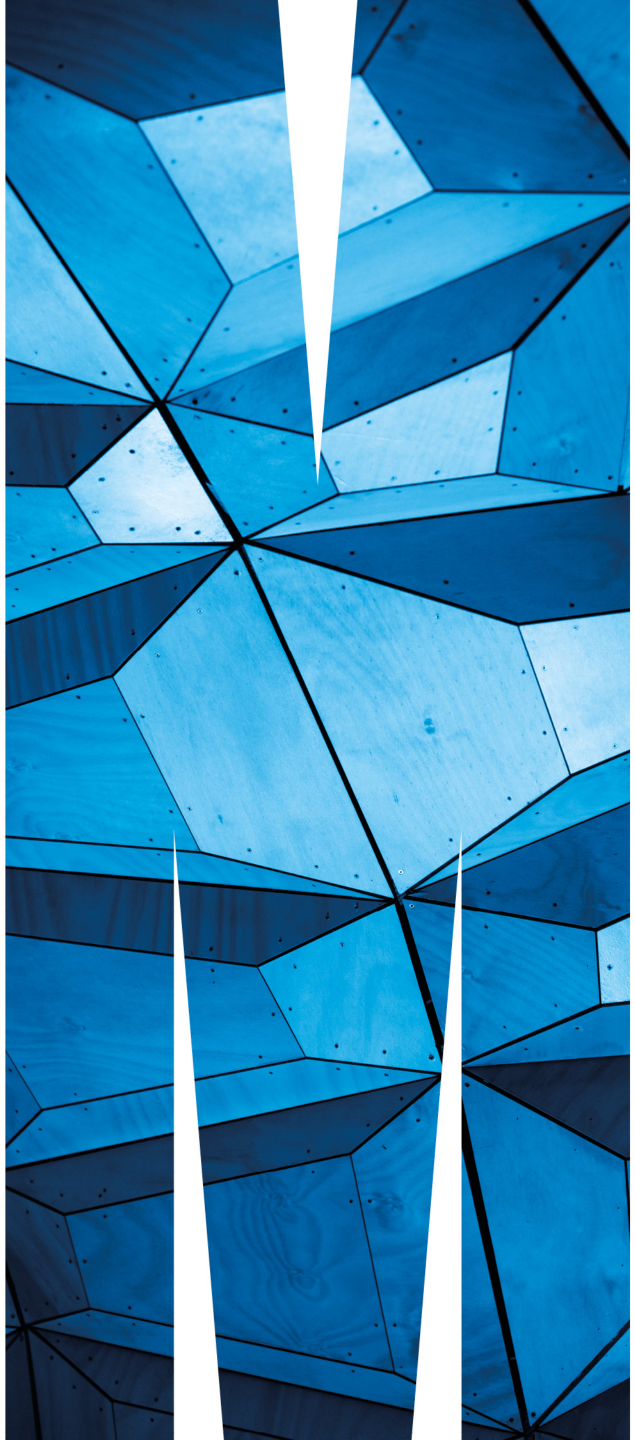
Scheduling Points

Checking points for theorem 2

These are the points in time at which ***any*** of the periodic tasks becomes active and may preempt the current execution if they have high enough priority..

For the example Task 3 the scheduling points are:
0, 100, 150, 200, 300, 350

At any scheduling point a task's contribution is its execution time multiplied by the number of times it has been scheduled to run starting at the critical instant (time 0)



Testing for Task 3:

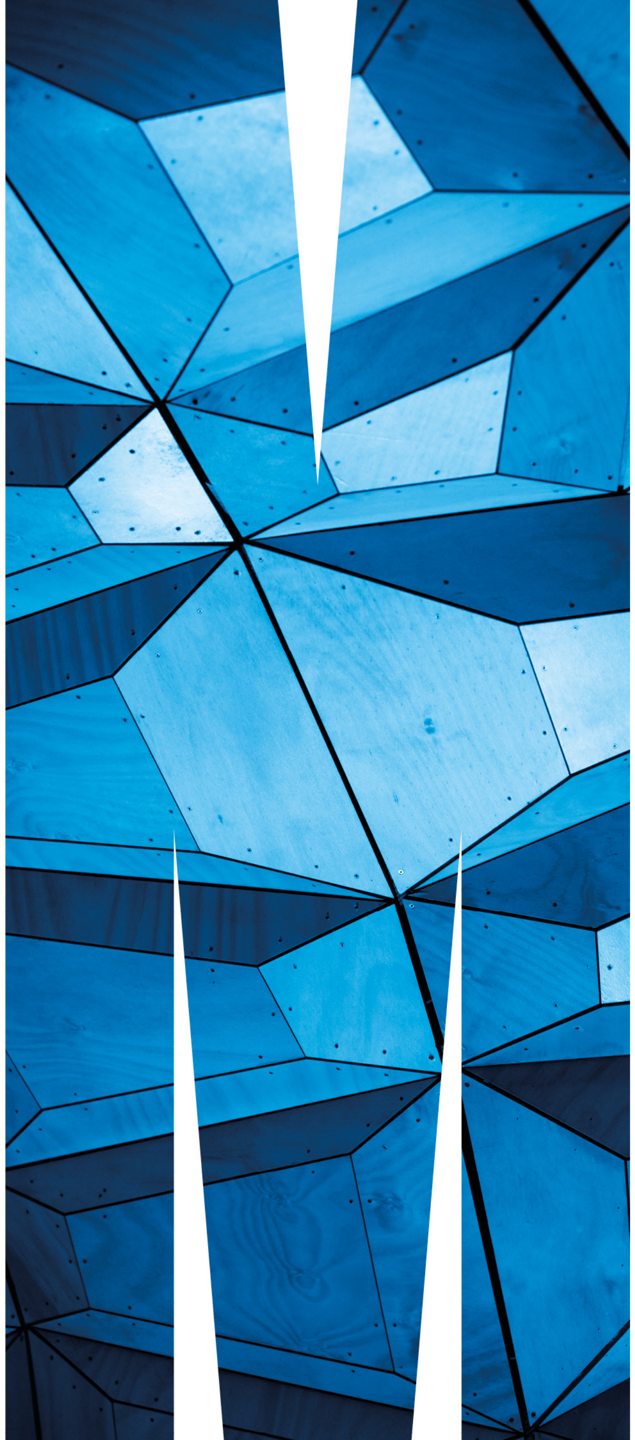
Is this condition satisfied at scheduling point 100?

Task1 + Task2 + Task3 contributions:

$$1x40 + 1x40 + 1x100 \leq 100$$

Obviously not as the summation for a single execution of each of the three tasks is 180.

Is it worth testing at 150?



Further Tests:

$$2x40 + 2x40 + 1x100 \le 200$$

$$260 \le 200 Fails$$

$$3x40 + 2x40 + 100 \le 300$$

$$300 \le 300 OK$$

This set is proven schedulable at time 300

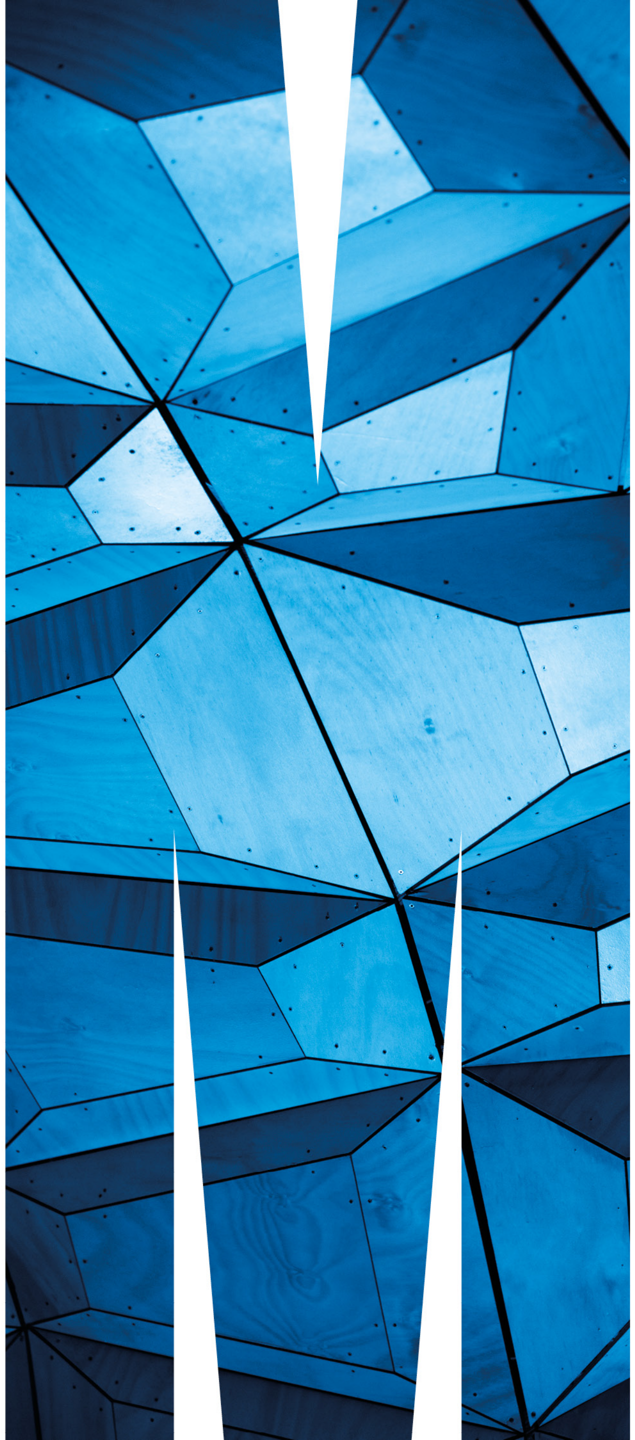
MONASH University

The Gotcha

$$4x40 + 3x40 + 100 \leq 350$$

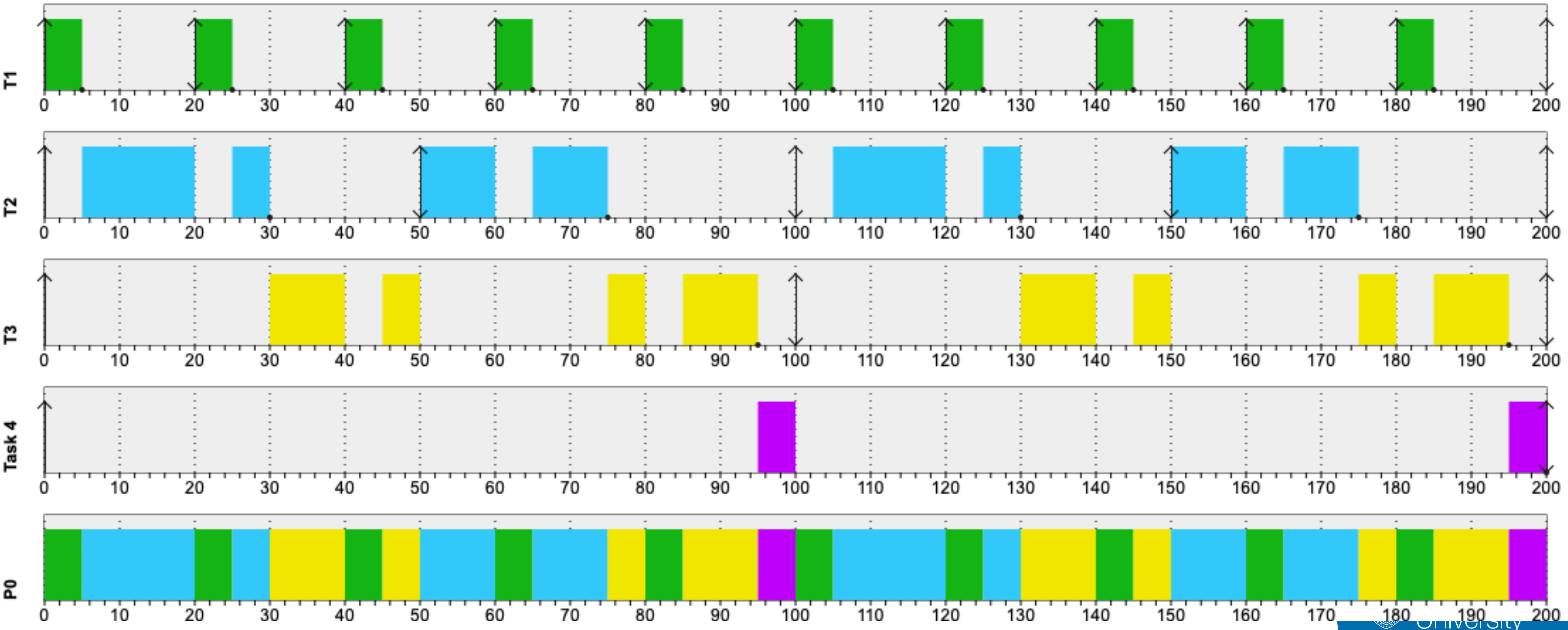
$$380 \leq 350 Fails$$

Testing the deadline for the task is not always giving the solution



Does the RMA work for higher utilisations?

```
Determine if the following task set is schedulable by RMA.
{(5,20)(20,50)(30,100)(10,200)}
```



What about EDF (Earliest Deadline First) for the same set?